ODIN

Macro-Assembleur
Editeur
Désassembleur
pour

MSX

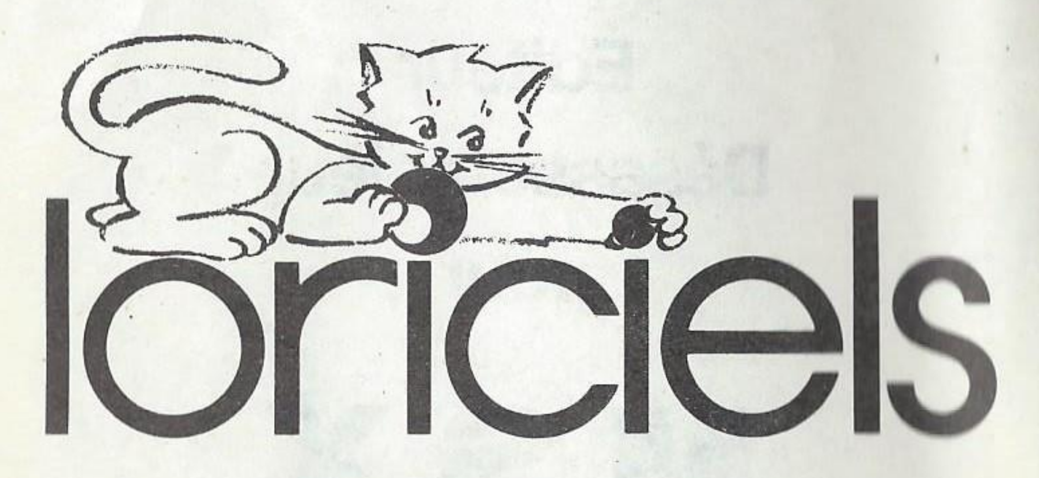
Bravo, vous venez d'acheter un système professionnel de développement en assembleur Z80 fonctionnant sur micro-ordinateur MSX.

Ce système est constitué de deux logiciels :

- L'EDITEUR ASSEMBLEUR contient deux parties :
Un EDITEUR puissant permettant l'écriture aisée des programmes en langage d'assemblage.
Un ASSEMBLEUR SYMBOLIQUE complet assurant la transcription du programme source en langage d'assemblage en code machine Z80.

- LE MONITEUR DESASSEMBLEUR sert à la mise au point des programmes en langage machine issus de l'Editeur Assembleur. Il permet entre autres le désassemblage du code Z80 et l'éxécution pas à pas en mémoire vive ou en mémoire morte.

Ce système de développement fonctionne sur micro-ordinateur MSX 16K, 32K ou 64K et permet l'utilisation de toute la mémoire vive disponible sur MSX 64K.



MONITEUR DESASSEMBLEUR MSX

1) Introduction

Ce logiciel a pour but de faciliter la mise au point des programmes en langage machine sur ordinateur MSX.
Il est entièrement écrit en langage machine pour assurer une plus grande efficacité et une plus grande souplesse d'utilisation.
Il occupe environ 6Koctets en mémoire vive et fonctionne sur tout ordinateur MSX 16K, 32K ou 64K.

Dans tout ce qui suit, les termes entre crochets ([et]) désignent des termes optionnels et les termes entre symboles d'inégalité (< et >) désignent des touches du clavier.

2) Chargement de la cassette

Pour charger le programme, placez la cassette dans votre magnétophone et tapez sur votre micro la commande suivante:

BLOAD"MONIT",R <RETURN>

Le programme se charge alors pendant quelques minutes et s'exécute automatiquement.
A coté du nom du programme s'affiche son implantation en mémoire sous la forme:
adrdeb - adrfin   où adrdeb et adrfin désignent les adresses en hexadécimal du premier octet et du dernier octet du Moniteur.
En bas de l'écran, apparaît le symbole > suivi du curseur invitant l'utilisateur à taper une commande.
Chaque fois que ce symbole sera affiché, cela indiquera que le Moniteur est en attente d'une commande.

3) Décomposition de l'écran

Le moniteur fonctionne en mode texte du processeur graphique (SCREEN 0). L'affichage se fait sur 24 lignes de 38 caractères.
Les 17 premières lignes de l'écran (lignes 0 à 16) sont destinées à visualiser le contenu des mémoires en hexadécimal (commande Memory) ou en mnémoniques Z80 (commande Desassemble).
Les 3 lignes suivantes (lignes 17 à 19) contiennent, soit la valeur des registres du Z80, soit la valeur des 10 points d'arrêt.
La ligne 20 visualise l'instruction placée à l'adresse contenue dans le compteur ordinal (registre PC).
Les lignes 21 et 22 sont destinées à saisir la ligne de commande tapée par l'utilisateur.
La dernière ligne (ligne 23) visualise les messages d'erreurs et les résultats de certaines commandes. Le contenu de cette ligne restera intact tant qu'aucune autre commande n'aura été tapée par l'utilisateur.

4) Le clavier et les touches de contrôle

Pour permettre l'utilisation du Moniteur, la gestion du clavier a été entièrement revue.

Toutes les touches du clavier ne génèrent que des caractères ASCII majuscules et non plus des instructions BASIC. L'appui de <SHIFT> et d'une autre touche permet de générer le petit symbole correspondant sur la touche.

La touche <BACKSPACE> ou la flèche gauche effacent le caractère situé juste avant le curseur.

La touche <RETURN> indique la fin de l'écriture d'une ligne de commande.

La touche <BREAK> (CTRL + STOP) permet d'interrompre certaines commandes exécutées par le Moniteur.

5) Commandes du Moniteur

Les commandes du Moniteur sont constituées d'un caractère qui est l'initiale du nom de la commande, suivi d'un certain nombre de paramètres.

Ces commandes peuvent contenir un nombre quelconque d'espaces qui ne sont pas pris en compte. De même les caractères situés après la commande ne sont pas pris en compte.

Voici la liste des commandes du Moniteur:

5.1. Breakpoint

Syntaxe: B — Affichage des points d'arrêt
Bn= — Suppression du point d'arrêt de numéro n et affichage des points d'arrêt
Bn=nnnn — Pose du point d'arrêt n à l'adresse nnnn et affichage des points d'arrêt

Le Moniteur permet de gérer jusqu'à dix points d'arrêt numérotés de 0 à 9. Un point d'arrêt peut être considéré comme un piège dans le programme. Lorsque le programme de l'utilisateur lancé par la commande Go atteint un point d'arrêt, il y a retour au Moniteur avec affichage des registres et de l'instruction suivante. A partir de ce moment, l'exécution pourra reprendre normalement (commande Go) ou pas à pas.

Pour réaliser ces points d'arrêt, le Moniteur place une instruction de saut au Moniteur à l'adresse de chacun des points d'arrêt, au moment de la commande de lancement (Go). Aussi, il faut tenir compte de ces trois octets ajoutés lors de la pose des points d'arrêt. Il ne faudra pas, en particulier, placer deux points d'arrêt distants de moins de trois octets. Un point d'arrêt situé en mémoire morte (adresse 0 à 7FFFH) sera totalement inefficace.

Lors du retour au Moniteur, les instructions de saut correspondantes à chacun des points d'arrêt seront remplacées par les octets originaux du programme de l'utilisateur.

L'affichage des points d'arrêt se fait dans les lignes 17 à 19 de l'écran. Chaque point d'arrêt est noté par un chiffre représentant son numéro suivi de l'adresse où il est placé. L'adresse zéro indique un point d'arrêt annulé.

Exemples: B6=9EDEH — Pose du point d'arrêt 6 à l'adresse 9EDEH
B0= — Suppression du point d'arrêt 0

L'affichage des points d'arrêt sur l'écran sera le suivant:

```
0:0000   1:0000   2:0000   3:0000
4:0000   5:0000   6:9EDE   7:0000
8:0000   9:0000
```

5.2. Calculate

Syntaxe: C expression

Cette commande permet d'évaluer une expression arithmétique constituée d'opérandes et d'opérateurs.

Le Moniteur reconnait les quatre opérateurs suivants:
+ addition
- soustraction
* multiplication
/ division

Une expression arithmétique contenant ces opérateurs est évaluée en tenant compte de la priorité des opérateurs * et / par rapport à + et -.

Ainsi 5+3*6 est évalué comme 5+(3*6).

Les opérandes sont des constantes numériques en décimal, en octal ou en hexadécimal.

Pour indiquer qu'un nombre est en octal ou en hexadécimal, il faut faire suivre ce nombre respectivement de la lettre "O" (pour Octal) ou "H" (pour Hexadécimal). Si aucune lettre ne suit le nombre ou si c'est la lettre "D", il sera considéré comme un nombre décimal.

Si une expression est incorrecte, le Moniteur affichera le message "Expression illegale" et s'il y a débordement des calculs, il affichera le message "Debordement".

Le résultat est affiché sur la dernière ligne de l'écran par les message "Resultat= " suivi de la valeur calculée de l'expression.

Exemples: C65*67H-2*34O donne 19EFH
C-33+57/2 donne FFFBH (-5)

Remarque: Dans toutes les commandes du Moniteur, il est possible d'utiliser une expression arithmétique de ce type pour chacune des valeurs ou des adresses qu'il demande.

5.3. Desassemble

Syntaxe: D[P]adr-deb[,adr-fin]

Désassemble en mnémoniques Z80 le programme situé entre les adresses adr-deb et adr-fin, bornes comprises.

Si P est spécifié, le résultat sort sur imprimante.

Sinon le désassemblage sort sur écran par pages de 17 lignes. Entre chaque page, le Moniteur attend que l'utilisateur tape sur une touche pour afficher la page suivante.

Si adr-fin est omis, l'adresse FFFFH est prise par défaut.

Il est possible d'interrompre le désassemblage en appuyant sur la touche <BREAK>.

Exemples: DP0,7FFFH — Désassemblage sur l'imprimante de la ROM
DA000H — Désassemblage sur écran du programme débutant en A000H

5.4. Enable/disable interrupts

Syntaxe: E

Cette commande permet de spécifier au Moniteur si les interruptions doivent être autorisées ou interdites lors des commandes d'exécution (commande Go et exécution pas à pas).

L'exécution de cette commande a pour effet de changer l'état de permission des interruptions. Au chargement du programme, les interruptions sont interdites. L'exécution de la commande E autorisera les interruptions et le message "Interruptions autorisees"

sera affiché. Une nouvelle exécution de cette commande interdira les interruptions et le message "Interruptions interdites" sera affiché.

5.5. Find

Syntaxe: F[adr-deb[,octet1[,octet2[,...]]]]

Recherche de la chaîne d'octets octet1, octet2,... en mémoire à partir de l'adresse adr-deb.

Si la suite d'octets est trouvée, le Moniteur passe en mode visualisation de mémoire (commande Memory) à la position du premier octet de la chaîne.

Si elle n'est pas trouvée, le message "Chaine non trouvée" est affiché.

Si la suite d'octets est omise dans la commande, le Moniteur prend la dernière chaîne recherchée.

Si l'adresse de début n'est pas donnée, la recherche commencera à partir de la dernière valeur de cette adresse, c'est à dire l'adresse de la précédente occurence de la chaîne plus un. Aussi, pour rechercher toutes les occurences d'une chaîne en mémoire, il suffira de donner la commande complète au départ pour rechercher la première occurence puis de taper la commande F seule pour les autres occurences.

La taille de la chaîne est limitée à 10 caractères. Si vous rentrez une chaîne plus longue, le message "Parametres incorrects" sera affiché.

Exemple: FA000H,56H+12,120   Recherche à partir de l'adresse A000H de la chaîne 62H, 0AH

5.6. Go

Syntaxe: G[adr]

Exécution du programme à partir de l'adresse adr.

Si adr est omise, le Moniteur prend la valeur du compteur ordinal (registre PC) comme adresse de lancement.

Tous les points d'arrêt sont placés sauf celui situé à l'adresse de lancement. C'est à l'utilisateur de s'assurer que son programme passera bien par un des points d'arrêt qu'il a posé. Si tout se passe bien, il y aura retour au Moniteur lors du passage sur un des points d'arrêt, avec visualisation de l'instruction suivante et des registres.

La commande Go interdit ou autorise les interruptions (voir commande Enable/disable interrupts).

Le Moniteur utilise la pile de l'utiliseur dans cette commande. Aussi, il faut que le pointeur de pile ne soit situé ni en mémoire morte, ni dans le Moniteur. Si ce n'est pas le cas, le message "Erreur Pile" est affiché et la commande n'est pas effectuée.

Exemple: G 746*68   Lancement du programme à l'adresse C628H

5.7. Indicator

Syntaxe: I

Visualise les différents indicateurs du registre F.

Les bits du registre F sont affichés les uns après les autres en commençant par le bit de poids fort. Les bits nuls sont symbolisés par "-". Les bits à un sont représentés par une lettre qui est l'initiale de leur nom ou par le symbole "1".

Les indicateurs sont résumés dans le tableau suivant:

Indicateur	Sign	Zero	Half/Carry	Parity	Add/Sub	Carry
Initiale	S	Z	H	P	N	C

Exemple: I   Si F vaut FFH, le résultat sera:
Indicateur= SZ1H1PNC

5.8. Load

Syntaxe: L[nom]

Charge le programme dont le nom est spécifié.

Le Moniteur charge le prochain programme de code machine qui se trouve sur la cassette et dont le nom commence par celui qui est donné dans la commande.

Si le nom est omis, le Moniteur charge le prochain programme de la cassette.

Lors du chargement, le nom du programme trouvé, son implantation en mémoire et son adresse de lancement s'affichent en bas de l'écran sous la forme :
adresse début - adresse fin - adresse de lancement

Un signal sonore annonce la fin du chargement.

Si une erreur de chargement se produit, le message "Erreur de chargement" sera affiché.

Vous pouvez interrompre le chargement en appuyant sur la touche <BREAK>.

Exemple: LTEST   Chargement du programme TEST

5.9. Memory

Syntaxe: M[P]adr-deb[,adr-fin]

Visualisation de la mémoire entre les adresses adr-deb et adr-fin.

Le Moniteur affiche le contenu des mémoires en hexadécimal et en ASCII par pages de 128 octets. En haut de la page sont affichés la première et la dernière adresse de la page. La colonne de gauche donne l'octet de poids faible de l'adresse du premier des octets visualisés en vis à vis. Les lignes centrales contiennent chacune 8 octets sous leur représentation hexadécimale à gauche et sous leur représentation ASCII à droite.

Si le paramètre "P" est spécifié, la sortie se fait sur imprimante.

Sinon l'affichage se fait sur l'écran et le Moniteur passe en mode modification.

En appuyant sur les touches "-" et "+", on commande l'affichage de la page précédente et de la page suivante.

Un curseur clignotant apparait sur l'écran. Les quatre flèches du clavier permettent de déplacer ce curseur. Pour modifier la mémoire, il faut positionner le curseur sur l'octet à modifier et inscrire sa nouvelle valeur en hexadécimal sur l'écran. Cette valeur sera automatiquement reportée en mémoire. La touche <RETURN> permet de revenir au mode commande du Moniteur.

Exemples: MP100H,1000   Affichage de la mémoire de 100H à 3E8H sur imprimante
M100H   Affichage de la page commençant en 100H

5.10. Place

Syntaxe: P adr

Reloge le Moniteur à partir de l'adresse adr.
L'adresse donnée doit être telle que le Moniteur tienne en mémoire vive (adr > 7FFFH) et qu'il n'écrase pas l'ancienne version de celui-ci. Sinon le message "Parametres incorrects" sera affiché et la commande ne sera pas exécutée. Cette commande ne fonctionne pas sur MSX 16K où l'espace mémoire disponible est insuffisant pour permettre la relocation.
L'exécution de cette commande dure environ une seconde. Une fois le Moniteur déplacé, on retourne au début de son exécution avec affichage du titre et du nouvel emplacement du Moniteur.

Exemple: PD000H   Reloger le Moniteur à l'adresse D000H

5.11. Register

Syntaxe: R   Affichage des registres
Rrrr=nnnn   Modification du registre rrr à la valeur nnnn et affichage des registres

rrr peut prendre les valeurs suivantes:
AF, BC, DE, HL, AF', BC', DE', HL', IX, IY, SP, PC

Exemples: RHL'=3C00H   Positionnement du registre HL' à 3C00H
RIY=564H   Positionnement du registre IY à 0564H

L'affichage des registres sur l'écran aura l'allure suivante:

```
 AF=0000    BC=0000   DE=0000    HL=0000
AF'=0000   BC'=0000  DE'=0000   HL'=3C00
 IX=0000    IY=0564   SP=FFFF    PC=0000
```

5.12. Save

Syntaxe: S[nom],adr-deb,adr-fin,adr-lan

Sauvegarde du programme en langage machine compris entre les adresses adr-deb et adr-fin bornes comprises et dont l'adresse de lancement est adr-lan, dans un fichier de code machine sur cassette portant le nom donné. Ce nom est limité à 6 caractères. Adr-fin devra être supérieur ou égal à adr-deb.
Ce fichier pourra être chargé par le commande BLOAD du BASIC, sauvegardé par la commande BSAVE et exécuté par la commande USR.
Le Moniteur affiche le message "Preparer la cassette" et attend que l'utilisateur tape sur une touche avant de lancer la sauvegarde.
Un signal sonore annonce la fin de la sauvegarde.

Exemple: SPROG,E000H,E000H+345H,E000H   Sauvegarde du programme PROG compris entre les adresses E000H et E345H et dont l'adresse de lancement est E000H

5.13. Type

Syntaxe : T[4 bits de poids faible du port A8]

Cette commande sera trés utile aux possesseurs d'une machine MSX 64K. Elle permet de sélectionner le bloc de mémoire situé entre les adresses 0000 et 7FFF.

Sur une machine MSX 64K ordinaire, les valeurs utilisables pour le paramètre de la commande T sont :

paramètre	mémoire 0000 à 3FFF	mémoire 4000 à 7FFF
0	ROM	ROM
3	RAM	ROM
12	ROM	RAM
15	RAM	RAM

Lors de l'exécution de cette commande, le moniteur affiche la valeur du port A8 utilisée sous la forme :
Port A8 = valeur
L'effet de cette commande se fera sentir sur toutes les commandes du moniteur utilisant la mémoire située entre les adresses 0000 et 7FFF. Il sera ainsi possible d'exploiter pleinement les 64K de RAM disponibles sur MSX 64K.

5.14. Verify

Syntaxe: V[nom]

Cette commande permet de vérifier si la dernière sauvegarde sur cassette est correcte, par comparaison entre les octets en mémoire et les octets enregistrés sur cassette.
Son fonctionnement est analogue à celui de la commande LOAD, mais le programme sur cassette est comparé avec celui présent en mémoire au lieu d'être chargé.

5.15. eXecute

Syntaxe: X[P]

Exécution pas à pas sans interruption entre chaque instruction.
La seule façon d'interrompre l'exécution est de taper sur la touche <BREAK>.
Si P est spécifié, la suite d'instructions exécutées est affichée sur l'imprimante.

Exemple: XP

5.16. Effacement de l'écran

Syntaxe: <SHIFT> + <CLS>

Cette commande provoque l'effacement de l'écran et le passage à l'écran de texte.

5.17. Changement du mode d'affichage

Syntaxe : <SHIFT> + <0> passage en SCREEN 0
<SHIFT> + <1> passage en SCREEN 1
<SHIFT> + <2> passage en SCREEN 2
<SHIFT> + <3> passage en SCREEN 3

Ces commandes seront trés utiles pour la mise au point de programmes travaillant en mode graphique 1, 2 ou 3.

6) Exécution pas à pas

6.1. Mode ordinaire

Syntaxe: <INS>

Cette commande effectue l'exécution pas à pas de la prochaine instruction, c'est à dire de l'instruction qui se trouve à l'adresse pointée par le compteur ordinal (registre PC). Cette instruction peut se trouver aussi bien en mémoire vive qu'en mémoire morte.

Après l'exécution de cette instruction, le Moniteur affiche la nouvelle valeur des registres et de l'instruction suivante.

De la même façon que dans la commande Go, cette commande interdit ou autorise les interruptions (voir commande Enable/disable interrupts).

6.2. <u>Mode sous-programme</u>

Syntaxe: <SUP> ou <DEL>

Cette commande est semblable à la précédente pour toutes les instructions ordinaires. Les instructions d'appel à un sous-programme sont, quand à elles, exécutées en une seule étape comme s'il s'agissait d'une seule instruction.

7) <u>Messages d'erreurs</u>

Les messages d'erreurs sont affichés sur la dernière ligne de l'écran et sont accompagnés d'un signal sonore. Le message reste affiché tant que l'utilisateur ne tape sur aucune touche.

Message d'erreur	Signification
Commande illegale	Commande inconnue du Moniteur
Parametres incorrects	Un ou plusieurs paramètres sont incorrects. Vérifiez la syntaxe.
Expression illegale	Expression arithmétique illégale
Debordement	Débordement dans un calcul arithmétique
Chaine non trouvee	Chaîne non trouvée dans la commande Find
Erreur de chargement	Erreur de chargement sur cassette
Erreur Pile	Le pointeur de pile contient une adresse incorrecte dans la commande Go

8) <u>Exemple d'utilisation du Moniteur</u>

Considérons le programme suivant qui efface tout l'écran de texte:

```
E000 F3          DI              ;Interdire interruptions
E001 21 00 00    LD    HL,0      ;Début adresse écran de texte
E004 CD DF 07    CALL  7DFH      ;Envoi de l'adresse au processeur
                                 ;video
E007 01 C0 03    LD    BC,3C0H   ;Taille de l'écran
E00A 3E 20       LD    A,20H     ;Caractère ASCII espace
E00C D3 98       OUT   (98H),A   ;Ecriture sur l'écran
E00E 0B          DEC   BC        ;Décrémenter compteur de boucle
E00F 78          LD    A,B       ;Test compteur nul ?
E010 B1          OR    C         ;
E011 20 F7       JR    NZ,E00A   ;Si non boucler
E013 FB          EI              ;Autoriser interruptions
```

L'équivalent BASIC serait du type :

```
10 FOR I = 0 TO &H3C0
20 VPOKE I,&H20
30 NEXT I
```

Implantez ce programme à partir de l'adresse E000H en tapant la commande ME000H.

Le Moniteur affiche la page mémoire comprise entre les adresses E000H et E07FH et positionne le curseur sur le premier octet.

Ecrivez le programme en hexadécimal en tapant le code de chacune des instructions:
F3 21 00 00 CD DF 07 01 C0 03 3E 20 D3 98 0B 78 B1 20 F7 FB

Posez un point d'arrêt à la fin du programme:
B0=E014H

Lancez l'exécution en tapant:
GE000H

Vérifiez que les valeurs des différents registres ont été modifiées.

EDITEUR/ASSEMBLEUR MSX

1) Introduction

Ce logiciel a pour but de faciliter l'utilisation du langage machine sur ordinateur MSX grâce à l'emploi des mnémoniques Z80 et des étiquettes symboliques.

Il comprend deux parties:

- Un éditeur de textes permettant d'écrire et de modifier facilement les programmes écrits en langage d'assemblage.
- Un assembleur effectuant la traduction du langage d'assemblage en langage machine et générant le code binaire sur cassette ou directement en mémoire.

Il fonctionne sur MSX 16K, 32K ou 64K, mais il ne trouve son plein emploi que sur MSX 32K ou 64K, où la taille mémoire disponible est suffisamment importante pour permettre l'écriture de programmes consistants.

Il est entièrement écrit en langage machine pour assurer une plus grande efficacité et une plus grande souplesse d'utilisation.

Il occupe environ 8Ko en mémoire vive et il est fourni sur une cassette protégée.

Dans tout ce qui suit, les termes entre crochets ([et]) désignent des termes optionnels et les termes entre symboles d'inégalité (< et >) désignent des touches du clavier.

2) Chargement de la cassette

Pour charger le programme, placez la cassette dans votre magnétophone et tapez sur votre MSX la commande suivante:

BLOAD"ASM",R <RETURN>

Le programme se charge alors pendant quelques minutes, puis il démarre automatiquement.

Le nom du logiciel et le message de copyright s'affichent en haut de l'écran. En dessous, apparaît le symbole > suivi du curseur invitant l'utilisateur à taper une commande.

Chaque fois que ce symbole sera affiché, cela indiquera que l'Editeur/Assembleur est en attente d'une commande.

3) Le clavier et les touches de contrôle

Pour permettre l'écriture des programmes en langage d'assemblage, la gestion du clavier a été revue.

Toutes les touches du clavier ne génèrent que des caractères ASCII et non plus des instructions BASIC.

La touche <BACKSPACE> efface le caractère situé juste avant le curseur.

La touche <TAB> provoque le saut du curseur sur la prochaine tabulation de l'écran. Les tabulations sont situées sur les colonnes 6, 13, 20, 27, 34. Elles sont utilisées en langage d'assemblage pour faciliter la lisibilité des programmes.

La touche <BREAK> (<CTRL>+<STOP>) permet d'interrompre n'importe quelle commande exécutée par l'assembleur.

La touche <RETURN> indique la fin de l'écriture d'une ligne de commande ou de texte.

La touche <SPACE> permet de geler momentanément l'affichage sur l'écran. Elle sera utilisée avec les commandes A et P de l'éditeur qui font défiler du texte, pour permettre à l'utilisateur d'examiner ce qui s'affiche sur l'écran. L'affichage normal reprendra par l'appui sur n'importe quelle touche sauf <BREAK>.

4) L'éditeur de textes

L'éditeur de textes permet la manipulation de 10000 lignes, constituées d'au plus 127 caractères ASCII, et numérotées de 0000 à 9999. L'accès à une ligne se fait par son numéro qui fait office de clé.

Les commandes de l'éditeur sont constituées d'un caractère qui est l'initiale du nom de la commande, suivi d'un certain nombre de paramètres qui peuvent être des numéros de lignes.

L'éditeur gère une ligne courante qui représente la dernière ligne manipulée. Cette ligne est généralement prise par défaut dans les diverses commandes si l'utilisateur omet un numéro de ligne. Elle peut être spécifiée également par l'emploi du symbole ".".

Par ailleurs les symboles "#" et "$" représentent respectivement la première et la dernière ligne du texte.

Les touches ↓ et ↑ permettent de commander respectivement l'affichage de la ligne suivante et de la ligne précédente du texte. Ces touches ne sont effectives que si elles sont tapées au début de la ligne de commande.

Toutes les commandes de l'éditeur peuvent être écrites en minuscules ou en majuscules et peuvent contenir un nombre quelconque d'espaces ou de tabulations qui ne seront pas pris en compte. De même, les caractères situés après la commande ne sont pas pris en compte.

Voici la liste détaillée des commandes de l'éditeur de texte:

4.1. Branch

Syntaxe: B adresse

Provoque un branchement du micro-processeur à l'adresse indiquée. Cette adresse peut être notée en décimal, en hexadécimal (suivie de "H") ou en octal (suivie de "O").

Exemples:

B33450	Branchement à l'adresse 33450 en décimal
B9000H	Branchement à l'adresse 9000 en hexadécimal
BD1D1H	Retour à l'assembleur avec effacement du programme en mémoire
BD1EAH	Retour à l'assembleur sans effacement du programme en mémoire

Cette commande sera particulièrement utilisée pour exécuter des programmes en langage machine obtenus par assemblage en mémoire. En plaçant une instruction JP 0D1EAH à la fin de votre programme, vous reviendrez à l'éditeur/assembleur à la fin de son exécution.

4.2. Copy

Syntaxe: C ligne1,[ligne2],[ligne3]

Recopie du groupe de lignes compris entre ligne1 et ligne2, bornes incluses, juste après la ligne3. Tout le texte est ensuite renuméroté avec un incrément de un.

Le message "Parametres incorrects" sera affiché si l'une des conditions suivantes se produit:

- ligne1 > ligne2
- ligne1 <= ligne3 <ligne2
- ligne3 n'existe pas

Si ligne2 est omise, l'éditeur prend la valeur ligne1 par défaut. Par conséquent seul ligne1 est copiée.

Si ligne3 est omise, la ligne courante est prise par défaut

Exemple: C30,150,710 Recopie des lignes 30 à 150, juste après après la ligne 710

4.3. Delete

Syntaxe: D [ligne1][,ligne2]

Destruction des lignes comprises entre ligne1 et ligne2, bornes comprises.

Si ligne1 > ligne2, le message "Parametres incorrects" est affiché.

Si ligne1 est omise, l'éditeur prend la ligne courante par défaut.

Si ligne2 est omise, il prend la valeur ligne1 par défaut.

Exemple: D100 Destruction de la ligne 100

4.4. Edit

Syntaxe: E [ligne]

Passage en mode édition sur la ligne spécifiée.

Si la ligne n'existe pas, le message "Ligne inexistante" est affiché.

Si la ligne est omise, l'éditeur prend la ligne courante par défaut.

En mode édition, vous pouvez déplacer le curseur sur la ligne à modifier grâce aux touches ← et →, détruire un caractère par <BACKSPACE> ou insérer des caractères en frappant la touche correspondante.

La touche <RETURN> provoque l'arrêt du mode édition avec affichage de la ligne modifiée.

Exemple: E Edition de la ligne courante

4.5. Find

Syntaxe: F [chaîne]

Recherche de la chaîne spécifiée dans le texte à partir de la ligne suivant la ligne courante, et affichage de la ligne contenant la chaîne.

Si la chaîne n'est pas trouvée, le message "Chaine non trouvee" est affiché.

Si la chaîne est omise, l'éditeur prend comme chaîne, la dernière chaîne recherchée. Ainsi, pour rechercher toutes les occurences d'une chaîne dans le texte, il suffit de faire un Fchaîne lorsque la ligne courante est la première ligne du texte, pour rechercher la première chaîne puis F pour rechercher les suivantes.

Exemple: FDEB Recherche de la chaîne DEB

Remarque: Il est possible de placer un caractère de tabulation dans la chaîne à rechercher. Ainsi la commande FDEB<TAB> permettra de rechercher la ligne où est définie l'étiquette DEB.

4.6. Hard

Syntaxe: H [ligne1][,ligne2]

Affichage sur imprimante du bloc de lignes compris entre ligne1 et ligne2 bornes comprises.

Si ligne1 est omise, la ligne courante est prise par défaut.
Si ligne2 est omise, ligne1 est prise par défaut.
Si ligne1 > ligne2, le message "Parametrès incorrects" est affiché.
Si ligne2 est omise et si ligne1 n'existe pas l'éditeur affiche la ligne suivante.

Exemple: H#,$ Affichage de tout le texte sur imprimante

4.7. Insert
Syntaxe: I [ligne1][,incrément]
Passage en mode insertion de texte à partir de la ligne1 si elle n'existe pas ou à partir de ligne1+incrément si elle existe.
En mode insertion, l'éditeur affiche successivement tous les numéros des lignes à insérer suivis du curseur invitant l'utilisateur à écrire la ligne. Le passage d'un numéro de ligne à un autre se fait par l'ajout de la valeur incrément.
Pour arrêter le mode insertion, tapez sur <BREAK>.
Le mode insertion sera également interrompu avec affichage du message "Plus de place entre les lignes", si la ligne à insérer a un numéro supérieur à la prochaine ligne du texte.
Si ligne1 est omise, la ligne courante est prise par défaut.
Si l'incrément est omis, l'éditeur prend la dernière valeur de l'incrément utilisée. Au départ l'incrément est fixé à 10.

Exemple: I30,1 Insertion à partir de la ligne 30 avec un incrément de 1

4.8. Load
Syntaxe: L [nom]
Charge le programme source dont le nom est spécifié.
L'éditeur charge le premier programme de la cassette dont le nom commence par celui qui est inscrit dans la commande. Ce programme se place à la suite de celui qui était en mémoire sans l'effacer.
Pour conserver une numérotation croissante des numéros de lignes, il y aura intérêt à exécuter la commande Number à la suite du chargement lors de la concaténation de deux programmes.
Si le nom est omis, l'éditeur charge le prochain programme de la cassette.
Lors du chargement, le nom du programme trouvé s'affiche sur l'écran. Il est possible d'interrompre le chargement en appuyant sur <BREAK>.
Si une erreur de chargement se produit, le message "Erreur de chargement" sera affiché.

Exemple: LTEST Chargement du programme TEST

4.9. Number
Syntaxe: N [ligne1][,[ligne2][,[incrément]]]
Renumérotation du texte source à partir de ligne1 qui prend le numéro ligne2 et jusqu'à la fin du texte, en utilisant un pas valant incrément entre chaque ligne renumérotée.
Si ligne1 > ligne2, le message "Parametres incorrects" est affiché.
Si ligne2 est inférieure à la ligne précédent ligne1, le message "Parametres incorrects" est affiché.
Si les paramètres de la commande sont tels que le numéro de la dernière ligne soit supérieur à 9999, le message "Numero de ligne trop grand" est affiché et la renumérotation n'est pas effectuée.
Si ligne1 est omise, l'éditeur renumérote tout le texte.

Si ligne2 est omise, la valeur ligne1 est prise par défaut.
Si incrément est omis, la dernière valeur de l'incrément est utilisée.

Exemple: N,10,10 Numérotation de tout le texte qui est placé à partir de la ligne 10 avec un incrément de 10

4.10. Print
Syntaxe: P [ligne1][,ligne2]
Affichage en continu sur l'écran du bloc de lignes compris entre ligne1 et ligne2.
La touche <SPACE> sera utile dans cette commande pour geler momentanément l'affichage.
Si ligne1 > ligne2, le message "Parametres incorrects" est affiché.
S'il n'y a pas de texte en mémoire le message "Pas de texte" est affiché.
Si ligne2 est omise et si ligne1 n'existe pas, l'éditeur affiche la ligne suivante.
Si ligne1 est omise, la ligne courante est prise par défaut.
Si ligne2 est omise, ligne1 est prise par défaut.
Si ligne1 et ligne2 sont omises, l'éditeur affiche la ligne courante et les 23 lignes suivantes du texte.

Exemple: P.,$ Affichage du texte à partir de la ligne courante et jusqu'à la fin

4.11. Replace
Syntaxe: R [ligne1][,incrément]
Effacement de ligne1 si elle existe et passage en mode insertion à partir de cette ligne avec l'incrément spécifié.
La syntaxe est identique à la fonction Insert.

Exemple: R,1 Effacement de la ligne courante et passage en mode insertion avec un incrément de un

4.12. Save
Syntaxe: S [nom]
Sauvegarde sur cassette du texte source en mémoire sous le nom spécifié.
L'éditeur affiche le message "Preparer la cassette". Préparez donc la cassette. Appuyez sur <RETURN> une fois que cela sera terminé.
Il est possible d'interrompre la sauvegarde en appuyant sur <BREAK>.

Exemple: Stest Sauvegarde du programme test

4.13. User
Syntaxe: U
Affichage de la taille occupée par le texte en mémoire et de la taille de la zone mémoire libre, sous la forme:
xxxxx Octets libres (Nombre d'octets libres)
yyyyy Octets utilises (Nombre d'octets occupés)

5) L'assembleur

5.1. Format d'une ligne assembleur

Une ligne assembleur doit se présenter sous la forme suivante, où les termes entre crochets sont optionnels.

```
[étiquette][TAB code-op[TAB opérandes]][TAB][;commentaires]
```

Elle peut être écrite en minuscules ou en majuscules. Toutes les lettres seront transformées en majuscules par l'éditeur, à l'exception des commentaires et des caractères situés entre apostrophes.

5.2. <u>Etiquette symbolique</u>

Le terme "étiquette" placé au début de la ligne assembleur représente une étiquette symbolique contenant au plus six caractères.

Le premier caractère ne peut être qu'une lettre ou un symbole dont le code ASCII est supérieur à 63. De plus, les chiffres sont acceptés dans les cinq autres caractères.

Une étiquette est utilisée pour représenter l'adresse d'implantation de la ligne assembleur dans le programme objet ou pour représenter une constante fixe (voir EQU).

Si une étiquette est incorrecte, l'assembleur affichera le message "Etiquette incorrecte".

Exemple d'étiquettes: LOOP1 LA_BAS @READ

5.3. <u>Tabulation</u>

Le terme "TAB" désigne un nombre quelconque mais non nul d'espaces ou de caractères de tabulation.

Pour faciliter la lisibilité des programmes en langage d'assemblage, il est conseillé d'utiliser pour TAB un seul caractère de tabulation qui n'occupe qu'un octet en mémoire.

5.4. <u>Code opératoire</u>

Le terme "code-op" de la ligne assembleur désigne le code d'une instruction Z80 ou le code d'une directive d'assemblage (voir ce paragraphe).

L'assembleur reconnait tous les codes du Z80 défini par ZILOG.

Si l'utilisateur emploie un autre code, l'assembleur affichera le message d'erreur "Code illegal".

Exemples: LD ADD LDIR ORG

5.5. <u>Opérandes</u>

Le terme "opérandes" décrit le mode d'adressage utilisé et les registres employés selon la syntaxe définie par ZILOG.

Si le mode d'adressage est interdit, l'assembleur affichera le message "Mode d'adressage illegal".

Exemples: A,(HL) HL,DE A,6

5.6. <u>Commentaires</u>

Les commentaires se placent à la fin de la ligne assembleur et sont précédés du symbole ";".

```
Exemples: 0010 ; Ceci est un commentaire
          0020        RRA   ;Autre commentaire
```

5.7. <u>Expressions arithmétiques</u>

Dans tous les opérandes numériques, il est possible d'utiliser une expression arithmétique.

Les quatre opérateurs arithmétiques que reconnait l'assembleur sont les suivants:

+ addition
- soustraction
* multiplication
/ division

Une expression arithmétique contenant ces opérateurs est évaluée en tenant compte de la priorité de opérateurs * et / par rapport à + et -.

Ainsi 5+3*6 est évalué comme 5+(3*6)

Les opérandes situées entre les opérateurs peuvenc être:

- Le symbole $ qui indique le compteur ordinal
- Une étiquette symbolique
- Une constante de 8 ou 16 bits sous forme ASCII entre apostrophes (Ex: 'Yt' vaut 5974H, 'k' vaut 6BH)
- Une valeur numérique en décimal, en octal ou en hexadécimal.

Pour indiquer qu'un nombre est en octal ou en hexadécimal, il faut faire suivre ce nombre respectivement de la lettre "O" (pour octal) ou "H" (pour hexadécimal). Si aucune lettre ne suit le nombre ou si c'est la lettre "D", il sera considéré comme un nombre décimal.

Les nombres hexadécimaux qui commencent par une lettre (A, B, C, D, E, F) doivent être précédés d'un zéro afin que l'assembleur les distingue d'une étiquette symbolique.

Si une expression est incorrecte, l'assembleur affichera le message "Expression illegale" et s'il y a débordement des calculs il affichera le message "Debordement".

```
Exemples d'expressions: 45H+3*41O/5-64D vaut 24
                        -34+'hY'*2+5    vaut 53397
```

5.8. <u>Directives d'assemblage</u>

Les directives d'assemblage sont des pseudo opérateurs utilisés pour agir sur le déroulement de l'assemblage ou pour générer des octets quelconques.

Voici la liste détaillée de toutes les directives d'assemblage reconnues par l'assembleur:

<u>DEFB ou DB</u>

```
[étiquette]TAB DEFB TAB expression1[,expression2[,... ]]
               DB
```

Génére un octet de valeur expression pour chacune des expressions qui suivent cette directive.

```
Exemple: 0010 OCTGEN  DEFB    3,56H,'g',0
         Cette ligne génère les octets suivants en hexadécimal:
         03 56 67 00
```

Remarque: Si une expression est vide, l'éditeur génère un octet zéro.
Ainsi: DB 3,,4 génère 03 00 04

<u>DEFM ou DM</u>

```
[étiquette]TAB DEFM TAB 'suite de caractères ASCII'
               DM
```

Génère un octet correspondant au code ASCII de chacun des caractères tapés. Cette directive sera donc utilisée pour définir des messages.

Pour inclure une apostrophe dans la chaîne de caractères il faut doubler cette apostrophe.

```
Exemple: 0010 MESSAG  DEFM    'L''hiver'
```

Cette ligne génère: 4C 27 68 69 76 65 72

DEFS ou DS

```
[étiquette]TAB DEFS TAB expression
               DS
```

Réserve un nombre d'octets égal à l'expression qui suit la directive. Cette directive sera donc utilisée pour définir des variables dans le programme.

```
Exemple: 0020 VAR1      DEFS      2
      défini une variable de 2 octets
```

DEFW ou DW

```
[étiquette]TAB DEFW TAB expression1[,expression2[,... ]]
               DW
```

Cette directive est identique à DEFB, mais elle agit sur des mots de deux octets au lieu d'agir sur des octets.

```
Exemple: 0040 WORDS    DEFW     45-4,,'gT',-1
      Cette ligne génère: 2900 0000 5467 FFFF
```

END

```
[étiquette]TAB END TAB [expression]
```

Indique la fin du texte à assembler. L'expression représente l'adresse d'exécution du programme. Celle-ci sera affichée à la fin de l'assemblage par le message:
xxxxx est l'adresse de lancement

La valeur par défaut de l'expression est zéro.

Si aucune directive END n'a été rencontrée lors de l'assemblage, le message "Pas de END" sera affiché.

```
Exemple: 0100          END      DEBUT+35H
```

EQU

```
étiquette TAB EQU TAB expression
```

Donne la valeur expression à l'étiquette.

Il n'est pas possible de définir plus d'une fois une étiquette par la directive EQU.

Les références aux étiquettes définies par EQU ne peuvent apparaitre qu'après la définition de ces étiquettes. Par contre les références aux autres étiquettes peuvent se faire n'importe quand. Le nom respect de cette règle entraîne la message "Symbole indefini".

```
Exemple: 1000 ICI      EQU      7F00H
      donne la valeur 7F00H à l'étiquette ICI
```

IF et ENDIF

```
[étiquette]TAB IF TAB expression
[étiquette]TAB ENDIF
```

Assemblage conditionnel

Si la valeur expression est nulle, toutes les instructions suivants la directive IF jusqu'à la directive ENDIF ne seront pas assemblées. Si l'expression est non nulle, cette directive est sans effet sur l'assemblage.

Il est possible d'utiliser jusqu'à 256 niveaux de directives IF imbriquées.

S'il y a plus de directives ENDIF que de directives IF, le message "ENDIF sans IF" sera affiché.

```
Exemple: 0100          IF        1-US
         0300          ENDIF
```

LIST

```
[étiquette]TAB LIST TAB ON
[étiquette]TAB LIST TAB OFF
```

LIST OFF arrête le listing sur l'écran ou l'imprimante et LIST ON le fait reprendre.

ORG

```
[étiquette]TAB ORG TAB expression
```

Défini l'implantation en mémoire des instructions qui suivent. Après l'exécution de cette directive, le code objet sera implanté à l'adresse donnée par expression.

```
Exemple: 1000          ORG        9000H
         implantation du code en 9000H
```

5.9. Commande d'assemblage

La commande "A" de l'éditeur permet d'appeler l'assembleur.
Syntaxe: A [nom][-option1[-option2[-...]]]

Cette commande provoque l'assemblage du texte en mémoire. Les options d'assemblage sont les suivantes:

- NL (No listing) Pas de listing
- LP (Line Printer) Sortie du listing sur imprimante
- WS (Write Symbol) Affichage de la table des symboles classée
- WE (Wait on Error) Attente à chaque erreur
- WO (Write Output) Sortie du binaire sur cassette en utilisant le nom spécifié
- IM (Into Memory) Assemblage en mémoire

Les options WO, WS et IM sont incompatibles

"nom" représente le nom du fichier sur cassette destiné à recevoir le code objet. Si ce nom dépasse 6 caractères, le message "Nom trop long" est affiché.

5.10. Résultat de l'assemblage

L'assembleur sort sur l'écran ou sur l'imprimante (option LP) succesivement chaque ligne du texte précédée de l'adresse d'implantation et du code généré. Si la ligne comporte des erreurs, les messages d'erreur seront affichés juste avant celle-ci. Dans l'option WE, l'assembleur attend que l'utilisateur frappe une touche chaque fois qu'une erreur est rencontrée.

A la fin de l'assemblage, le nombre d'erreurs est affiché et l'adresse d'exécution du programme. Si l'option IM a été spécifiée, l'assembleur charge le programme en mémoire à condition que l'adresse d'implantation se situe en haut de mémoire, après le texte de

l'utilisateur (voir la carte mémoire de l'assembleur). Si l'option WO a été spécifiée, l'assembleur affiche le message "Preparer la cassette". Il génère alors le code objet sur un fichier cassette qui pourra être lu par la commande BLOAD "nom" du BASIC ou par le commande L du Moniteur. Si le programme est trés gros, il se peut que l'assembleur soit obligé de fractionner le programme objet en plusieurs fichiers qui porteront le nom spécifié suivi d'un numéro. Si le programme s'implante à une adresse inférieure à 8000H sur un MSX 64K, il ne pourra être lu qu'avec la commande L du Moniteur.

6) Messages d'erreur

6.1. Messages d'erreur émis par l'éditeur

Lorsque l'éditeur détecte une erreur, il affiche le message d'erreur correspondant sur l'écran et interrompt immédiatement la commande. Les messages d'erreur qu'il peut afficher sont les suivants:

MESSAGE	SIGNIFICATION
Commande illegale	Commande inconnue
Parametres incorrects	La commande n'est pas utilisée avec des paramètres corrects
Ligne inexistante	Appel d'une ligne inexistante
Numero de ligne trop grand	Numéro de ligne supérieur à 9999
Plus de place entre les lignes	Le numéro de la prochaine ligne à insérer est supérieur au numéro de la ligne suivante du texte
Pas assez de memoire	Il n'y a plus de place en mémoire pour insérer du texte
Chaine non trouvee	Chaîne de caractères non trouvée (commande Find)
Increment nul	L'incrément utilisé est nul
Pas de texte	Demande d'affichage d'une ligne alors qu'il n'y a pas de texte en mémoire
Erreur de chargement	Erreur de chargement
Nom trop long	Nom du fichier sur cassette dépassant 6 caractères

6.2. Messages d'erreur émis par l'assembleur

Lorsque l'assembleur détecte une erreur dans une ligne du texte, il affiche juste avant celle-ci le, ou les messages d'erreur sur l'imprimante (option LP) ou sur l'écran même si l'option NL à été demandée.

Suivant la gravité de l'erreur, il génère ou ne génère pas de code pour cette ligne. Dans les deux cas, l'assemblage n'est pas interrompu. Il ne s'arrêtera que lors de la rencontre de la directive END ou à la fin du texte si cette directive est absente. Seul l'erreur "Debordement table des symboles" arrête l'assemblage.

Erreurs mineures (code généré)

MESSAGE	SIGNIFICATION
Fin de ligne illegale	Des caractères illégaux apparaissent juste aprés les opérandes
Expression illegale	Expression arithmétique incorrecte
Debordement	Débordement dans une expression arithmétique
Debordement de champ	Opérande numérique supérieure à 255 ou à 127 respectivement dans un adressage d'octet ou dans un adressage relatif
Branchement trop eloigne	Branchement relatif hors des limites
Etiquette incorrecte	Etiquette incorrecte
Symbole indefini	Référence à une étiquette non définie
Definition multiple	Définition multiple d'une étiquette
Symbole multiplement defini	Référence à une étiquette définie plusieurs fois
Pas de END	Absence de directive END

Erreurs majeures (code non généré)

MESSAGE	SIGNIFICATION
Code illegal	Code opératoire illegal
Mode d'adressage illegal	Mode d'addressage illegal
Adresse incorrecte	Adresse trop basse lors d'un assemblage en mémoire. Le code est affiché sur l'écran, mais n'est pas implanté en mémoire
Trop de IF imbriques	Utilisation de plus de 256 niveaux de IF imbriqués
ENDIF sans IF	Directive ENDIF non associée à une directive IF
Debordement table des symboles	Il n'y a pas assez de place en mémoire pour contenir la table des symboles. L'assemblage est interrompu.

7) Exemple

```
D1EA           0010 WBOOT   EQU     OD1EAH     ;Adresse de retour
C800           0020         ORG     OC800H     ;Implantation du prog.
C800 F3        0030 DEBUT   DI                 ;Interdire IT.
C801 3EFO      0040         LD      A,0FOH     ;Modifier le port A8
C803 D3A8      0050         OUT     (0A8H),A   ;pour utiliser la ROM
C805 210000    0060         LD      HL,0       ;Adresse debut ecran
C808 CDDF07    0070         CALL    7DFH       ;Adresse au TMS 9918
C80B 3E20      0080         LD      A,32       ;Espace
C80D D398      0090 LOOP    OUT     (98H),A    ;Afficher caractere
C80F 3C        0100         INC     A          ;Caractere suivant
C810 20FB      0110         JR      NZ,LOOP    ;Boucler si <> 255
C812 FB        0120         EI                 ;Remettre IT pour clav.
C813 CD9F00    0130 LA      CALL    9FH        ;Lecture clavier
C816 28FB      0140         JR      Z,LA       ;Pas de touche tapee
C818 C3EAD1    0150         JP      WBOOT      ;Retour à l'assembleur
C800           0160         END     DEBUT      ;Fin

00000 Erreur(s)
51200 est l'adresse de lancement
```

Ce petit programme provoque un remplissage de l'écran par le jeu de caractères ASCII, puis il attend que l'utilisateur appuie sur une touche pour revenir à l'éditeur/assembleur.

L'équivalent BASIC serait le suivant :

```
10 PRINT CHR$(11);         Debut de l'ecran
20 FOR I = 32 TO 255       Boucle d'ecriture des caracteres
30 PRINT CHR$(I);          Affichage caractere
40 NEXT I                  Fin de boucle
50 IF INKEY$ = "" GOTO 50  Attente frappe d'une touche
```

8) <u>Carte mémoire</u>

```
0000  |-----------| |
      |           | |
      |           | | Buffer de texte sur MSX 64K (51K)
      |           | |
8000  |-----------| |     |
      |           | |     | Buffer de texte sur MSX 32K (19K)
C000  |-----------| |     |     | Buffer de texte sur MSX 16K (3K)
      |-----------| V     V     V
      |Table des  |
      | symboles  |
      |-----------|
      |Libre pour |
      | code Z80  |
D000  |-----------|
      |ASSEMBLEUR |
F380  |-----------|
      |Variables  |
      |   MSX     |
FFFF  |-----------|
```

ERRATA

Afin d'être compatible avec l'ensemble des MSX,
nous avons dû en dernière minute
modifier le paragraphe "5.17." du manuel.
Nous vous prions de bien vouloir
prendre connaissance des nouvelles données
de ce paragraphe.

5.17. <u>Changement du mode d'affichage</u>

Syntaxe : <F1> passage en SCREEN 0
<F2> passage en SCREEN 1
<F3> passage en SCREEN 2
<F4> passage en SCREEN 3